Kapees: A New Tool for Standard Cell Placement

Sameer Pawanekar[1,⋆], Kalpesh Kapoor[2], and Gaurav Trivedi[1]

[1] Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati, India
{p.sameer,trivedi,kalpesh}@iitg.ernet.in
[2] Department of Mathematics
Indian Institute of Technology Guwahati, India

Abstract. We consider the well-known problem of efficient cell placement on a fixed die. We investigate minimization of half perimeter that is required for a design that in turn results into minimal routed wire length and thus wire delay. We describe a new method, *Kapees*, for large scale standard cell placement. Our technique is based on recursive partitioning of placement circuit which is modeled as a hypergraph. It uses partitioning during the global placement phase and a greedy approach is followed to reduce the wire length during detailed placement phase. Our results show a significant improvement in comparison to Cadence Encounter's Amoeba and Capo tools by 9% and 5%, respectively.

1 Introduction

Standard Cell Placement is a well studied problem over several years. The objective of standard cell placement is to find coordinates of all the standard cells in a netlist in such a way that the wire length connecting them is minimum. The wire length is modeled as Half Perimeter wire length (HPWL) which can be defined as sum of all the perimeters of the smallest bounding box enclosing each net of the design. There are four broad approaches to solve this problem: 1. Min-cut [1–3], 2. Simulated annealing [4], 3. Analytic [5], and 4. Force directed [6]. Although both academic and commercial tools for placement are available, there is a scope for improvement because of inherent complexity of the problem [7]. This is also apparent from the ISPD placement contests held in the recent past in which none of the placers dominated across the entire benchmark set. A comparative study has also shown that the current state of the art is far from optimal [8].

Our approach to solve this problem is based on partition driven placement. Net cut objective follows wire length objective at initial hierarchical levels. At later hierarchical levels, net cut objective no longer follows wire length objective. This is when the number of cells are less than say 10. After this step detail placement (DP) follows. The existing placers such as Capo [1], Dragon [2] and feng shui [3] use different approaches during detailed placement.

⋆ Sameer Pawanekar is currently a senior engineer with SiConTech, Bangalore. He is enrolled as a part-time Ph.D. student at IIT Guwahati.

M.S. Gaur et al. (Eds.): VDAT 2013, CCIS 382, pp. 66–73, 2013.

The tool feng shui does k-way partitioning and legalization and carries out the detailed placement by a branch and bound method [3]. In addition to traditional partitioning with hMetis, feng shui also uses large-scale k-way partitioning by iterative deletion to obtain initial terminal propagation information. Both Capo and Dragon uses a top-down hierarchical partitioning approach. Capo uses the hypergraph partitioner tool MLPart [9]. Dragon performs bin swapping for net cut optimization and low temperature annealing in its global and detailed placement phases, respectively [2].

Partition driven placement tools rely on terminal propagation to a large extent. This technique is capable of obtaining better cuts with regard to placement. Work in [10] relies heavily on the terminal propagation technique. It helps the partitioner to gather the characteristics of cut nets and derive hence good wire length results. The tool NTUPlace [11] proposes a terminal propagation method which was a generalized version of a method presented in Theto [10].

We present a new method, *Kapees*, for the cell placement which has two steps (a) partitioning in global placement phase, and (b) low temperature annealing in detailed placement phase, followed by greedy cell swapping. We compare our HPWL results with two existing tools: Cadence Encounter's Amoeba and Capo [12].

The rest of the paper is organized as follows. In Section 2, we present the flow of our algorithm and our global placement strategy. Section 3 presents low temperature annealing algorithm. Finally, experimental results and comparison with three other tools and the conclusion are given in Section 4.

2 Global Placement Phase

Our placement tool works in two phases – Global Placement Phase and Detailed Placement Phase. We describe these phases in detail below. The former phase involves partitioning of hypergraph using hMetis package [13]. We recursively bipartition the circuit and perform the arrangement. An arrangement is a step in which coordinates are assigned to the bins in all possible ways. We continue to partition until an optimum level is reached. We aim to use bisection because of the fact that we got better partitioning and wirelength results in case of bisection compared to using k-way partitioning and our partitioning approach is not aimed at obtaining the slicing of floorplan In the next phase, we use a simulated annealing algorithm with multiple objectives which is then followed by a greedy algorithm for Half Perimeter Wire Length (HPWL) reduction. In global placement phase we recursively bipartition the given circuit. We first partition the circuit horizontally followed by vertically partitioning the two sub-circuits that are obtained in the previous step. In the second step, these four partitioned circuits are bisected. This continues until an optimum level is reached which is defined as $\log(\textit{number of rows})$. If the number of rows is four, we can recursively bipartition the circuit up to two levels. At the end of partitioning phase we are left with $4^{\log(\textit{number of rows})}$ sub-circuits that we refer to as bins. Each bin typically has between five to ten cells. During global phase all the cells in a

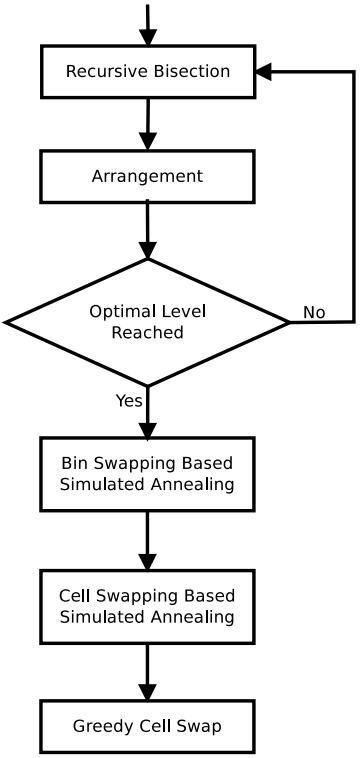

Fig. 1. Flow of Kapees

bin have the same coordinates. At every level an arrangement of the bins in all possible locations is performed. For the first level, entire circuit forms a bin and is placed at the center. In the second level, there are four bins and 24 possible ways of arranging the four bins in four locations. In the third level we have $4 \times 4 = 16$ bins. Here each bin in the third level is arranged in 24 ways within the region of its parent bin.

The existing placers such as [2] and [10] typically carry out terminal propagation. In terminal propagation, when the cells of circuit is partitioned into two bins, dummy cells are introduced into each bins. These dummy cells are included in external (cut) nets. When the two bins are possible candidate for further partitioning, external nets carry information that they are connected to a bin externally, and hence they do not get cut. As a result better wire length is obtained. However, in our experiment with terminal propagation we did not find noticeable improvement in the wire length resulting from terminal propagation scheme. So we do not perform terminal propagation but instead swap and move cells within bins. This is equivalent to improve upon the cuts obtained at Global placement phase. We discuss Cell Swapping in section 3.2.

3 Detailed Placement Phase

In Detailed placement phase we perform bin swapping based simulated annealing and cell swapping based simulated annealing followed by greedy cell swapping.

3.1 Bin Swapping Based Simulated Annealing

In bin swapping phase, we use a simulated annealing algorithm (see Algorithm 1) with two objectives. These are (A) reduction of wire length, and (B) balancing row width, respectively.

To implement Multi-Objective Simulated Annealing, the objective function, C, has to be equal to a sum of two objective functions $C = \alpha A + \beta B$. It is difficult to determine the ratio α and β as these values may not be the same for different designs. We overcome this problem by selecting the moves in such a way that any move that causes an imbalance is not accepted. We swap coordinates of the bins to achieve HPWL reduction along with the condition that row imbalance should not occur. The minimization function, $F(x)$, is the x-direction HPWL estimate and is given by Equation (1) from [14], where n_H and e_H are the node and edge sets of the hypergraph H, respectively.

$$F(x) = \sum_{e_k \in e_H} \max_{\forall i,j \in n_H} |x_i - x_j| \tag{1}$$

Algorithm 1. Multi Objective Simulated Annealing

```
while init_temp ≥ final_temp do
    P = e^{δWL/init_temp}
    Perturb                                  ▷ Will cause swapping of bin coordinates
    if δ WL ≤ 0 and row_imbalance ≤ 0 then
        Accept the perturbation
    else
        if random_number ≤ P and row_imbalance ≤ 0 then
            Accept the perturbation
        else
            Revert the perturbation
        end if
        if sufficient_number_of_perturbations then
            Decrease init_temp
        end if
    end if
end while
```

3.2 Cell Swapping Based Simulated Annealing

We do not perform terminal propagation therefore it is important for us to improve the cuts obtained during global placement phase. This is achieved by

performing cell swapping between the bins. In this step all cells within a bin are placed on top of each other, that is, there exists overlap between the cells. All the cells of a bin have the same coordinates. After completion of cell swapping between the bins, all the cells are spread out to remove overlap. Spreading is placing of the cells of the bins adjacent to each other in such a way that there is no overlap between them. It is important that the wire length obtained after spreading out all the cells must correlate with the changes that are done during cell swapping phase. Cell swapping and cell movement between the bins is studied by [15] and implemented in Dragon version 2.1. They attempt to achieve a bin width balance by swapping of cells. By doing this they achieve a correlation between the wire length of spread out cells and wire length during cell swapping. We tried two approaches for doing annealing based on cell swapping.

Algorithm 2. Perturbation Method 1

Randomly Select Two Bins in sufficiently close vicinity
Select one Cell from Each Bin
Calculate Incremental Cost = Cost1
Swap the Cell coordinates
Calculate Incremental Cost = Cost2
Return $\delta WL = Cost1 - Cost2$

Algorithm 3. Perturbation Method 2

Randomly Select Two Bins in sufficiently close vicinity
Select one Cell from Each Bin
Calculate Incremental Cost = Cost1
Swap the Cell coordinates and arrange all Bins to the right of the selected Bins separated by a distance equal to their widths
Calculate Incremental Cost = Cost2
Return $\delta WL = Cost1 - Cost2$

In the first approach we did not move the bin location ie, all the bins have fixed coordinates. Here the bin coordinates are not changed and there exists even spacing between the bins. Here we randonly select the bins which are within a distance of $numberofrows/3$ bins. For us this method does not correlate well with the spread out wire length.

In the second approach Figure 2, bins are placed at a distance equal the their widths and we shift the bins towards right by the difference in cell widths. Let Cell A belongs to Bin A and Cell B belongs to Bin B, then after swapping the cells, all the bins to the right of Bin A will be moved by a difference amount (Width(cellB)-width(cellA)) where as all the bins to the right of Bin B Bin B will be moved to right with a difference amount (Width(cellA)-width(cellB)). If we spread out the cells after the first method, we get worse wire length. This is

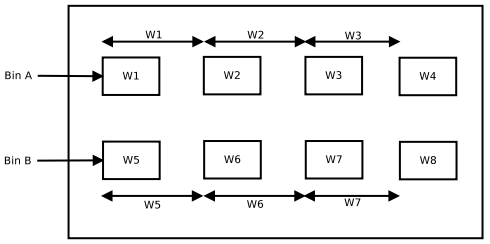

Fig. 2. Method 2 for Perturbation

because the estimates for bin width separation are not correct. We do not aim to obtain a balance for bin widths but only for row widths. In the second method the wire length during cell swapping stage correlates well with spread out cost.

3.3 Greedy Cell Swapping

In this stage, initially, all the cells for the bins are spread out. Then we randomly pick two cells in sufficiently close vicinity and see if swapping their coordinates with each other can reduce the wire length. The swapping is done in such a way that there does not exist any overlap between the adjacent cells. All the cells to the right of the selected cells are spread out.

Table 1. Characteristics of IBM version 2 benchmarks

Circuits	Cell Count	Net Count	Rows	core utilization
IBM01_easy	12028	11753	132	85.12%
IBM01_hard	12028	11753	130	88.00%
IBM02_easy	19062	18688	153	90.42%
IBM02_hard	19062	18688	149	95.28%
IBM07_easy	44811	44681	233	89.95%
IBM07_hard	44811	44681	226	95.30%
IBM08_easy	50672	48230	243	90.03%
IBM08_hard	50672	48230	236	95.16%
IBM09_easy	51382	50678	246	90.24%
IBM09_hard	51382	50678	240	95.12%
IBM10_easy	66762	64971	321	90.22%
IBM10_hard	66762	64971	313	95.08%
IBM11_easy	68046	67422	281	90.11%
IBM11_hard	68046	67422	273	95.33%

4 Experimental Results and Conclusion

We selected the IBM version 2.0 benchmarks to test our tool. These benchmarks have cell counts ranging from 12028 to 68046. Most of the current state of art

Table 2. HPWL comparison for IBM version 2 benchmarks

Circuits	Amoeba (A)	Capo (C)	Kapees (K)		
	$\times 10^6$	$\times 10^6$	$\times 10^6$	K/A	K/C
IBM01_easy	58.0278	55.8873	51.7759	0.89	0.92
IBM01_hard	57.6793	55.1354	51.456	0.89	0.93
IBM02_easy	165.906	158.743	146.253	0.88	0.92
IBM02_hard	164.541	156.048	145.224	0.88	0.93
IBM07_easy	373.388	368.7	354.121	0.94	0.96
IBM07_hard	359.586	355.686	353.964	0.98	0.99
IBM08_easy	406.834	387.502	352.928	0.86	0.91
IBM08_hard	395.116	379.512	363.081	0.91	0.95
IBM09_easy	341.532	317.126	311.554	0.91	0.98
IBM09_hard	338.188	321.029	321.057	0.94	1.00
IBM10_easy	605.746	636.689	601.824	0.99	0.94
IBM10_hard	642.065	629.543	624.721	0.97	0.99
IBM11_easy	521.413	481.637	478.191	0.91	0.99
IBM11_hard	514.758	476.332	472.878	0.91	0.99
			Average	0.91	0.95

placers have reported their work on these benchmarks. For our experiments and HPWL comparison we obtained the tool Cadence Encounter's Ameoba (version 9.1) and Capo (version 8.8) [12] from its respective websites.

The implementation of our tool, Kapees, for large scale standard cell placement, is done in C language. The experiments are performed on 1.5 GHz, 32-bit, 2GB RAM, Intel dual core machine running Ubuntu, a variant of GNU-Linux, as operating system. We compared our tool with Capo as they are run on the same machine, whereas Amoeba run results are extracted on a different machine.

The run time of our tool is 10 times more when compared to that of Capo for designs with cells less than 2000. For the design IBM01_easy, run time of Capo is 21 seconds, whereas, runtime of Kapees is 428 seconds, which means kapees is 22 times slower than Capo for this design. For the design IBM12_hard, run time of Capo is 205 seconds, whereas, runtime of Kapees is 9084 seconds, which means kapees is 44 times slower than Capo for this design. We observed that the runtime of Kapees increases with the growth of number of cells. This is primarily due to the use of simulated annealing technique wherein we perform HPWL calculation for each iteratve move. We plan to improve run-time in future.

As reported in Table 2, the experiments show that the half perimeter wire lengths obtained for these designs by Kapees are on an average 9% and 5% less than that obtained from Amoeba and Capo, respectively. We get superior results over other simulated annealing techniques because of our contribution as the method2. Instead of using terminal propagation, we use method2 to improve the already obtained cuts during global placement.

References

1. Caldwell, A.E., Kahng, A.B., Markov, I.L.: Can recursive bisection alone produce routable, placements? In: Proceedings of Design Automation Conference, pp. 477–482 (2000)
2. Wang, M., Yang, X., Sarrafzadeh, M.: Dragon2000: standard-cell placement tool for large industry circuits. In: Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 260–263 (2000)
3. Agnihotri, A., Yildiz, M.C., Khatkhate, A., Mathur, A., Ono, S., Madden, P.H.: Fractional cut: improved recursive bisection placement. In: Proceedings of International Conference on Computer Aided Design (ICCAD), pp. 307–310 (November 2003)
4. Sechen, C., Sangiovanni-Vincentelli, A.: The timberwolf placement and routing package. IEEE Journal of Solid-State Circuits 20(2), 510–522 (1985)
5. Kahng, A.B., Wang, Q.: Implementation and extensibility of an analytic placer. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24(5), 734–747 (2005)
6. Cong, J., Xie, M.: A robust detailed placement for mixed-size ic designs. In: Proceedings of Asia and South Pacific Conference on Design Automation, pp. 188–194 (January 2006)
7. Shahookar, K., Mazumder, P.: VLSI cell placement techniques. ACM Computing Surveys 23(2), 143–220 (1991)
8. Chang, C.C., Cong, J., Romesis, M., Xie, M.: Optimality and scalability study of existing placement algorithms. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 23(4), 537–549 (2004)
9. Alpert, C.J., Caldwell, A.E., Kahng, A.B., Markov, I.L.: Hypergraph partitioning with fixed vertices (VLSI CAD). IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 19(2), 267–272 (2000)
10. Selvakkumaran, N., Karypis, G.: Theto - a fast and high-quality partitioning driven global placer. Technical report, in university of minnesota - computer science and engineering technical reports (2003)
11. Chen, T.C., Hsu, T.C., Jiang, Z.W., Chang, Y.W.: Ntuplace: a ratio partitioning based placement algorithm for large-scale mixed-size designs. In: Proceedings of the 2005 International Symposium on Physical Design, ISPD 2005, pp. 236–238. ACM, New York (2005)
12. Capo: Tool, `http://vlsicad.eecs.umich.edu/BK/PDtools/tar.gz/Placement-bin/` (accessed 17 January 2013)
13. Karypis, G., Kumar, V.: Multilevel k-way hypergraph partitioning. In: Proceedings of 36th Design Automation Conference, pp. 343–348 (1999)
14. Kennings, A., Markov, I.: Analytical minimization of half-perimeter wirelength. In: Proceedings of Design Automation Conference (ASP-DAC), pp. 179–184 (June 2000)
15. Yang, X., Choi, B.K., Sarrafzadeh, M.: A standard-cell placement tool for designs with high row utilization. In: Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp. 45–47 (2002)